

DOCKET NO.: MSFT-0742/177739.01

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:

Nicholas P. Wilt, Sameer A. Nene and
Joseph S. Beda III

Confirmation No.: 2352

Application No.: 10/039,036

Group Art Unit: 2195

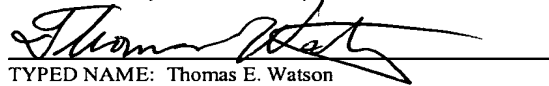
Filing Date: January 4, 2002

Examiner: Syed J. Ali

For: Methods And System For Managing Computational Resources Of A Coprocessor In
A Computing System

DATE OF DEPOSIT: May 16, 2006

I HEREBY CERTIFY THAT THIS PAPER IS BEING
DEPOSITED WITH THE UNITED STATES POSTAL
SERVICE AS FIRST CLASS MAIL, POSTAGE PREPAID,
ON THE DATE INDICATED ABOVE AND IS
ADDRESSED TO THE COMMISSIONER FOR PATENTS,
P.O. BOX 1450, ALEXANDRIA, VA 22313-1450.


TYPED NAME: Thomas E. Watson
REGISTRATION NO.: 43,243

MS Appeal Brief - Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**APPEAL BRIEF TRANSMITTAL
PURSUANT TO 37 CFR § 41.37**

Transmitted herewith in triplicate is the APPEAL BRIEF in this application with respect to
the Notice of Appeal received by The United States Patent and Trademark Office on
March 16, 2006.

- ☐ Applicant(s) has previously claimed small entity status under 37 CFR § 41.37.
- ☐ Applicant(s) by its/their undersigned attorney, claims small entity status under 37
CFR § 1.27 as:
- ☐ an Independent Inventor
 - ☐ a Small Business Concern
 - ☐ a Nonprofit Organization.

DOCKET NO.: MSFT-0742/177739.01

PATENT

- ☐ Petition is hereby made under 37 CFR § 1.136(a) (fees: 37 CFR § 1.17(a)(1)-(4) to extend the time for response to the Office Action of _____ to and through _____ comprising an extension of the shortened statutory period of _____ month(s).

	SMALL ENTITY		NOT SMALL ENTITY	
	RATE	FEE	RATE	FEE
<input checked="" type="checkbox"/> APPEAL BRIEF FEE	\$250	\$	\$500	\$500.00
<input type="checkbox"/> ONE MONTH EXTENSION OF TIME	\$60	\$	\$120	\$
<input type="checkbox"/> TWO MONTH EXTENSION OF TIME	\$225	\$	\$450	\$
<input type="checkbox"/> THREE MONTH EXTENSION OF TIME	\$510	\$	\$1020	\$
<input type="checkbox"/> FOUR MONTH EXTENSION OF TIME	\$795	\$	\$1590	\$
<input type="checkbox"/> FIVE MONTH EXTENSION OF TIME	\$1080	\$	\$2160	\$
<input type="checkbox"/> LESS ANY EXTENSION FEE ALREADY PAID	minus	(\$)	minus	(\$)
TOTAL FEE DUE		\$0		\$500.00

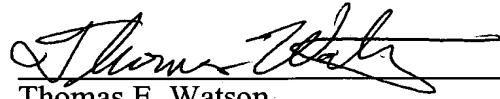
- ☒ The Commissioner is hereby requested to grant an extension of time for the appropriate length of time, should one be necessary, in connection with this filing or any future filing submitted to the U.S. Patent and Trademark Office in the above-identified application during the pendency of this application. The Commissioner is further authorized to charge any fees related to any such extension of time to Deposit Account 23-3050. This sheet is provided in duplicate.
- ☐ A check in the amount of \$.00 is attached. Please charge any deficiency or credit any overpayment to Deposit Account No. 23-3050.
- ☒ Please charge Deposit Account No. 23-3050 in the amount of \$500.00. This sheet is attached in duplicate.

DOCKET NO.: MSFT-0742/177739.01

PATENT

- ☒ The Commissioner is hereby authorized to charge any deficiency or credit any overpayment of the fees associated with this communication to Deposit Account No. 23-3050.

Date: May 16, 2006



Thomas E. Watson

Registration No. 43,243

Woodcock Washburn LLP
One Liberty Place - 46th Floor
Philadelphia PA 19103
Telephone: (215) 568-3100
Facsimile: (215) 568-3439

© 2006 WW



DOCKET NO.: MSFT-0742/177739.01

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Nicholas P. Wilt, Sameer A. Nene,
Joseph S. Beda, III

Confirmation No.: 2352

Application No.: 10/039,036

Group Art Unit: 2195

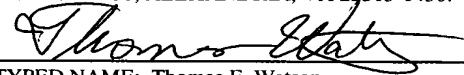
Filing Date: January 4, 2002

Examiner: Syed J. Ali

For: Methods and System for Managing Computational Resources of a Coprocessor in a Computing System

DATE OF DEPOSIT: May 16, 2006

I HEREBY CERTIFY THAT THIS PAPER IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE AS FIRST CLASS MAIL, POSTAGE PREPAID, ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO THE COMMISSIONER FOR PATENTS, P.O. BOX 1450, ALEXANDRIA, VA 22313-1450.


TYPED NAME: Thomas E. Watson
REGISTRATION NO.: 43,243

Mail Stop Appeal-Brief Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPELLANTS' BRIEF PURSUANT TO 37 C.F.R. § 41.37

This brief is being filed in support of Appellants' appeal from the final rejections of claims 1-23, 27-49 and 52-74 under 35 U.S.C. § 103, dated November 16, 2005. A Notice of Appeal was timely filed on March 16, 2005.

05/19/2006 RFEKADU1 00000021 233050 10039036
01 FC:1402 500.00 DA

TABLE OF CONTENTS

I.	REAL PARTY IN INTEREST	3
II.	RELATED APPEALS AND INTERFERENCES	4
III.	STATUS OF CLAIMS	5
IV.	STATUS OF AMENDMENTS	6
V.	SUMMARY OF CLAIMED SUBJECT MATTER.....	7
VI.	GROUND OF REJECTION TO BE REVIEWED ON APPEAL	9
VII.	ARGUMENT	10
1.	Claims 1-13, 16-23, 27-39, 42-49, 52-64 and 67-74 are not Rendered Obvious over Anderson in view of Duruoz under 35 U.S.C. § 103.....	10
A.	Summary of Prosecution History to Date	10
B.	Background concerning the Law of Obviousness	12
C.	Any System that Combines the System of Anderson and the System of Duruoz has Inoperability Issues that cannot be Resolved	13
D.	Appellants Detailed Specification Concerning Command Buffer Scheduling Provides Further Evidence that the Invention is not Obvious.....	17
2.	Claims 14-15, 40-41 and 65-66 are not Rendered Obvious over Anderson in view of Duruoz and further in view of Hendler.....	20
3.	Conclusion	21
VIII.	CLAIMS APPENDIX.....	22
IX.	EVIDENCE APPENDIX.....	30
X.	RELATED PROCEEDINGS APPENDIX	31

I. REAL PARTY IN INTEREST

The real party in interest is Microsoft Corporation of Redmond, Washington. This case has been assigned to Microsoft Corporation, as evidenced by the assignment recorded on January 4, 2002 at Reel 012484, Frames 0794-0797, a correction thereto having been recorded on April 1, 2002 at Reel 012826, Frames 0600-0607.

II. RELATED APPEALS AND INTERFERENCES

There are presently no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-23, 27-49 and 52-74 are pending in this case, all of which stand finally rejected in the Final Office Action, dated November 16, 2005. This appeal is from the rejection of claims 1-13, 16-23, 27-39, 42-49, 52-64 and 67-74 under 35 U.S.C. § 103, and also from the rejection of claims 14-15, 40-41 and 65-66 under 35 U.S.C. § 103.

IV. STATUS OF AMENDMENTS

Claims 1-5, 9, 11, 13, 15-16, 23, 27, 29-31, 35, 37, 39, 41-42, 49, 52, 54-56, 60, 62, 64, 66-67 and 74 were amended on September 9, 2005 to clarify that command buffers include commands.

Claims 24-26 and 50-51 were also canceled on September 9, 2005, without prejudice.

No other amendments have been made in the present application.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Against this background, Appellants now turn to the claims that are the subject of this appeal. In this regard, Appellants herein summarize the independent claims in order to provide an overview of the subject matter involved in this appeal.

Representative claim 1, for instance, relates to a method for controlling computational resources of coprocessor(s) in a host computing system having a host processor, including controlling the coprocessor(s) with commands of command buffers submitted to the coprocessor(s) by a host processor of the host computing system, transmitting, by the coprocessor(s), data back to the host computing system in response to commands in the command buffer(s) and scheduling the transmission of the commands of the command buffers by a managing object included in the host computing system, wherein the computational resources of the coprocessor(s) are simultaneously available to multiple applications instantiated on the host computing system. Support for claim 1 can be found in the specification at least at page 4, line 11 to page 5, line 7, and specifically page 4, lines 19-26. See also page 6, lines 10-15. See also page 23, lines 3-13. See also page 26, lines 5-19. See also page 28, lines 12-20. See also, page 36, line 28 to page 37, line 7.

Similarly, representative claim 27 relates to computer readable media for controlling the computational resources of coprocessor(s) in a host computing system having a host processor including a managing object for controlling the coprocessor(s) of the computing system with command data of command buffers submitted to the coprocessor(s) by a host processor of the host computing system and for scheduling the transmission of the command data and means for transmitting, by the coprocessor(s), data back to the host computing system in response to command data of at least one command buffer of the command buffers, whereby the computational resources of the coprocessor(s) are simultaneously available to multiple applications instantiated on the host computing system. Support for claim 27 can be found in the specification at least at page 4, line 11 to page 5, line 7, and specifically page 4, line 27 to page 5, line 7. See also page 6, lines 10-15. See also page 23, lines 3-13. See also page 26, lines 5-19. See also page 28, lines 12-20. See also, page 36, line 28 to page 37, line 7.

Lastly, representative claim 52 relates to a computing device having a host processor for controlling the computational resources of coprocessor(s) in a host computing system, comprising a managing object for controlling the coprocessor(s) of the host computing system with command data of command buffers submitted to the coprocessor(s) by the host

processor of the host computing system and for scheduling the transmission of the command data of the command buffers and means for transmitting, by the coprocessor(s), data back to the host computing system in response to command data in at least one command buffer of the command buffers whereby the computational resources of the coprocessor(s) are simultaneously available to multiple applications instantiated on the host computing system. Support for claim 52 can be found in the specification at least at page 4, line 11 to page 5, line 7, and specifically page 4, line 27 to page 5, line 7. See also page 6, lines 10-15. See also page 23, lines 3-13. See also page 26, lines 5-19. See also page 28, lines 12-20. See also, page 36, line 28 to page 37, line 7.

Also, the block diagram of Fig. 4 (e.g., as described on page 36, line 16 to page 37, line 7) provides further support for the above methods, computer readable media and computing devices. As described in that embodiment, the command buffer scheduler 404 (“scheduler”) and kernel driver 405 work together in kernel mode to dispatch command buffers to the hardware 406 (the scheduler 404 decides which command buffer should be dispatched, while the kernel driver 405 instructs the hardware 406 to dispatch a command buffer at the request of the scheduler).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Whether claims 1-13, 16-23, 27-39, 42-49, 52-64 and 67-74 are obvious over U.S. Patent No. 5,577,250 (Anderson) in view of U.S. Patent No. 6,487,642 (Duruoz) under 35 U.S.C. § 103(a).

2. Whether claims 14-15, 40-41 and 65-66 are obvious over Anderson in view of Duruoz and further in view of U.S. Patent No. 6,473,777 (Hendler) under 35 U.S.C. § 103(a).

VII. ARGUMENT

1. Claims 1-13, 16-23, 27-39, 42-49, 52-64 and 67-74 are not obvious over Anderson in view of Duruoz.

A. Summary of Prosecution History to Date

To aid in understanding the relevant issues, Appellants review the state of the prosecution history as of the date of the present appeal concerning the alleged combinability of Anderson and Duruoz in forming the outstanding rejection under 35 U.S.C. § 103 (the subject of the present appeal). In the first Official Action, dated July 12, 2005, with respect to the obviousness rejection based on Anderson and Duruoz, it was stated:

It would have been obvious to one of ordinary skill in the art to combine Anderson and Duruoz since buffering commands instead of requiring immediate execution allows the host processor to continue in other tasks without waiting for the coprocessor to complete its commands. This allows other tasks with hard deadlines to be completed on time, creating a pipelined system of processing that allows more tasks to be serviced in the same period of time.

After Appellants traversed the rejection under § 103 in a first response, in a Final Official Action, dated November 16, 2005, it was stated that:

...Even if Applicants accept ... that Anderson does not teach away from the combination with Duruoz, it is conceded ... that valid motivation to combine is required ... for a prima facie case of obviousness. In the original Office Action, the knowledge generally available to a person having ordinary skill in the art was relied upon. This motivation is still believed valid, and reiterated [in the above quoted paragraph from the Official Action, dated July 12, 2005]. Nonetheless, to make it explicitly clear just how obvious this combination is, attention is directed to the Abstract of Duruoz, which discusses several advantages of using a command buffer.

Generally, these advantages are similar to those relied upon as being within the general knowledge of a person of ordinary skill originally relied upon by the Examiner. These advantages include: (1) the host need not wait for the command to be executed; (2) sorting commands so that time-critical commands are executed appropriately; (3) scheduling nonexclusive commands at the appropriate time; (4) allowing for prioritization of nonexclusive commands, etc. (Duruoz, Abstract) Thus, there is ample motivation to combine Anderson and Duruoz.

In this case, Appellants respectfully submit that consideration of the advantages of using command buffers as set forth in Duruoz is not relevant to the overall inquiry of whether one of ordinary skill in the art would be led to substitute Duruoz's command buffers for Anderson's DSP Task Units. Abstracting the point, whether a particular component has

advantages when used as that particular component is not relevant to the question of whether such particular component is substitutable for Anderson's DSP Task Units. In this regard, Appellants traversed the outstanding rejection on the basis that Anderson and Duruoaz do not appear to provide any suggestion for the interchangeability of command buffers (Duruoaz) and DSP Task Units (Anderson) such that an operative system would result. After Appellants traversed the final rejection, the Advisory Action, dated February 13, 2006, stated that:

Applicants' arguments center on the alleged lack of interchangeability between Anderson and Duruoaz. Applicant alleges that Anderson has extensive requirements for implementing tasks that are not compatible with the teachings of Duruoaz. First, it should be noted that the portion of Anderson cited by Applicants relates to the specifications of a DSP Task Unit, rather than the actual implementation of a DSP Task Unit. A closer reading of Anderson and Duruoaz actually reveals that the references are very much combinable.

Anderson discusses creating tasks for a DSP to execute by way of a data structure stored in a linked list. The data structure includes the relevant variables or operands, input/output buffers, etc. needed for executing the task (Col. 8, lines 24-29). DSP Task Units are created dynamically and organized in a linked list (Col. 8 lines 24-26). However, Anderson notes that special steps are required for handling the storage allocations associated with a Task Unit, suggesting a need for a mechanism that handles the allocation/deallocation of tasks.

Duruoaz provides such a mechanism in the form of a command buffer that also has the benefit of allowing tasks/commands to be received at the coprocessor so that the host processor can continue with other work while waiting for the coprocessor to complete the task. The command buffer is shown at Fig. 5 of Duruoaz, and is disclosed as a data structure for storing commands. Thus, the command buffer supports a task or command unit in the form of a data structure, which is perfectly combinable with the DSP Task unit object described by Anderson. Moreover, Duruoaz shows that the data structure should include the necessary operands for executing the task, while Anderson teaches inclusion of the necessary "variables" as part of the DSP Task Unit. Since Anderson and Duruoaz have been shown to be combinable, and Applicants present no other arguments regarding the deficiency of the combination thereof, the claims stand rejected.

Following the Advisory Action, Appellants herein respectfully urge reconsideration of the combinability of Anderson and Duruoaz so that the remaining rejection under 35 U.S.C. § 103 be withdrawn and allowability of the claims confirmed.

B. Background on the Law of Obviousness under 35 U.S.C. § 103

Appellants' position is that Anderson and Duruoz are not properly combinable when Federal Circuit precedent concerning the standard of obviousness is considered in the context of the art of record. Accordingly, Appellants first provide a review of Federal Circuit case law on the topic of obviousness so as to present the proper framework for analysis of the issues relating to the present appeal.

In this regard, to prevent the improper use of hindsight based on knowledge of the invention to defeat patentability of the invention, in order to be availed of the provisions of 35 U.S.C. § 103, a motivation to combine the references that create the case of obviousness must be shown. In other words, reasons that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements from the cited prior art references for combination in the manner claimed must be shown. There are three possible sources for a motivation to combine references: the nature of the problem to be solved, the teachings of the prior art, and the knowledge of persons of ordinary skill in the art. *In re Rouffet*, 149 F.3d 1350, 1357 (Fed. Cir. 1998).

The issue of obviousness was recently revisited by the Federal Circuit in *In re Kahn*, Docket No. 04-1616 (March 22, 2006), in which the Court gave reminder of the general framework that:

[t]o reject claims in an application under section 103, an examiner must show an un rebutted prima facie case of obviousness On appeal to the Board, an applicant can overcome a rejection by showing insufficient evidence of prima facie obviousness or by rebutting the prima facie case with evidence of secondary indicia of nonobviousness. *In re Rouffet*, 149 F.3d 1350 (Fed. Cir. 1998)

and specifically that:

“A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant.” *In re Gurley*, 27 F.3d 551, 553 (Fed. Cir. 1994).

In addition, it has been long established that where references taken in combination produce a “seemingly inoperative device,” they are said to teach away because one of ordinary skill in the art would not be led to create inoperable devices. *In re Spinnoble*, 405 F.2d 578 (CCPA 1969). Similarly, if the combined product would not have the properties sought by the Applicant, the references are also said to teach away from forming the combination. *In re Caldwell*, 319 F.2d 254 (CCPA 1963).

Moreover, in making any obviousness determination, “a prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention.” (MPEP § 2141.02) Thus, for instance, one may not view isolated parts of a reference in a vacuum, while ignoring other parts, for purposes of an obviousness determination.

C. Any System that Combines the System of Anderson and the System of Duruoz has Inoperability Issues that cannot be Resolved.

It is agreed that Anderson does not teach or suggest Appellants claimed invention at least because Anderson fails to teach or suggest command buffers. However, Duruoz is cited in combination with Anderson for its description of the use of command buffers and various advantages of using command buffers as stated in Duruoz.

However, Appellants respectfully submit that the systems of Anderson and Duruoz, when viewed in their entirety would lead one of ordinary skill in the art away from combining the references, and that one of ordinary skill in the art, without the benefit of hindsight, would not be led to shoehorn Duruoz’s command buffers into the DSP Task Unit system of Anderson. In this respect, as discussed above in connection with the law of obviousness, references must be viewed as a whole, i.e., no portions may be disregarded.

Accordingly, Appellants respectfully submit that the combination of Anderson and Duruoz teaches away from the present invention due to lack of interchangeability of the pre-defined DSP Task Unit structure of Anderson and the command buffers of Duruoz. In this regard, as described in more detail below, all of Anderson’s DSP Task Units are required to adhere to a pre-defined particular structure that is not believed to be generalizable in the manner suggested. For instance, some of these very particular requirements are reproduced as follows (with highlights in bold and underlining):

When a host application program utilizes a DSP coprocessor, a DSP Task is constructed ... by making macro calls to the DSP Manager via the Application Programming Interface (API). Using these macro calls, a task structure is created and a desired DSP Task Unit inserted into the task structure. Other macro calls are used to activate, query status and deactivate a DSP Task. Col. 8, lines 14-23.

The task structure is typically comprised of a task identifier, **associated flags**, and **a pointer to a linked list of DSP Task Units**. DSP Task Units comprise the DSP code, variables, input buffer, output buffer or other information for creating the DSP Task which is executed on the DSP coprocessor. Col. 8, lines 24-29.

Next, a new (empty) DSP Task structure is created, step 307. This involves the allocation of memory for storing the task data structure. Resources from DSP Task Units are then allocated and inserted into the DSP Task structure from storage, step 308. Information stored in the Task Unit from the TUDL is used in conjunction with application directives via the API to construct the DSP Task. Information concerning the DSP Task Units are provided to the host programmer from the DSP Task Unit Specification Document. Once DSP Task Unit insertion is completed, the DSP Task is then placed into a DSP Task run list for execution by the DSP coprocessor, step 309. When the DSP Task is inserted into the DSP Task run list, it is inactive, i.e. it is not flagged for execution. The DSP Task is then activated for execution, step 310. This would involve setting an active/inactive control flag associated with the task. Col. 8, line 61 to Col. 9, line 9.

The creation of a DSP Task Unit starts with the development (i.e. writing and debugging) of a desired DSP program, step 501.... First, a new Task Unit structure is created, step 502. This will result in the allocation of memory for the program unit structure and insertion of information into the header field. The DSP program code, data, variables, etc. are inserted into the DSP Task Unit structure, step 503. Next, the DSP Task Unit is prepared for used by a DSP Host Application or Client by compiling, and/or assembling and linking, step 504. Finally, the constructed Task Unit is placed in storage, e.g. a disk. Col. 9, line 55 to Col. 10, line 2.

In essence, Appellants feel that these particular requirements have not been factored into an overall analysis that presumes that mere mention of command buffers and their advantages, as disclosed in Duruoz, means that those command buffers can be squeezed into the very particular structure disclosed in the system of Anderson under 35 U.S.C. § 103.

For instance, Anderson describes that the task structure into which a task unit is inserted is typically comprised of a task identifier, associated flags, and *a pointer to a linked list of DSP Task Units*, implying that the task structure organizes task units via a linked list data structure that links task units. However, a linked list of memory objects is implemented by linking an item in the linked list to (a reference to a memory location of) the next item. In short, DSP task units of Anderson are specific data structures (including flags) and they are arranged in a specific manner (linked list). Command buffers are not arranged as linked lists, and there is no suggestion that command buffers can operably be arranged as linked lists in the manner taught by the system of Anderson. Thus, Duruoz' command buffers cannot be said to be substitutable for Anderson's linked list of DSP task units.

For another example, to create a new task structure, Anderson first describes a first step of allocating memory for the new task structure, and then describes a second step of

allocating resources from DSP Task Units and inserting them into the DSP Task structure from storage (step 308). However, no comparable two-step memory allocation techniques for utilizing command buffers exist and there is no teaching or suggestion in Anderson or Duruoz of how such two-step memory allocation techniques could be implemented for command buffers. Accordingly, one of ordinary skill in the art would not be led to combine Anderson and Duruoz due to memory allocation incompatibilities.

For another example, Anderson describes setting “active/inactive” flags for enabling and disabling the DSP Task Units in conjunction with other information contained in the DSP Task Unit data structure. Col. 9, lines 1-9. However, command buffers do not operate in this manner of setting “active/inactive” flags, and thus the setting “active/inactive” flags as described in Anderson is not consistent with utilizing Duruoz’ command buffers. There is thus an inoperability issue that would lead one of ordinary skill in the art away from implementing Anderson’s “active/inactive” flags for enabling or disabling Duruoz’s command buffers.

At bottom, Appellants thus respectfully submit that Anderson’s Task Units possess a lot of structure that is entirely inconsistent with implementing command buffers. For yet another example, Anderson discloses that when a new Task Unit structure is created, it results in the allocation of memory for the program unit structure and the insertion of information into the header field of the new Task Unit structure. At such time, the DSP program code, data, variables, etc. are inserted into the DSP Task Unit structure (step 503). Next, the DSP Task Unit is prepared for use by a DSP Host Application or Client by compiling, and/or assembling and linking (step 504). Simply put, command buffer architectures do not require a client to perform such compiling and/or assembling and linking. Thus, one of ordinary skill in the art would be led away from substituting the command buffers of Duruoz with the Task Units of Anderson because the specific techniques mandated by Anderson for preparing Task Units cannot be ported to command buffers. Accordingly, it is believed that such disclosure in Anderson would lead a person of ordinary skill in the art away from combining Duruoz with Anderson.

Still further, with respect to Duruoz’s command buffers, Duruoz is understood to disclose storing the last 16 executed commands in a DRAM command done buffer if acknowledgment is required by the host processor (See Col. 6, lines 57-63). A similar effect could not operably be incorporated into the linked list system of Anderson, and there is no teaching or suggestion in either Anderson or Duruoz of how such an effect could be

incorporated into the linked list system of Anderson. Accordingly, such disclosure in Duruoz would lead a person of ordinary skill in the art away from combining Anderson with Duruoz. Thus, there are two way inoperabilities that appear to be unresolved, i.e., Anderson and Duruoz form a “seemingly inoperative device” leading a person of ordinary skill in the art away from forming such a combination.

In sum, the extensive requirements imposed on the Task Unit structures of Anderson cannot be accommodated simultaneously with the command buffers described in Duruoz. Duruoz describes no such ways, or interchangeabilities between the DSP Task Unit and command buffer systems, and neither does Anderson. While “knowledge of one of ordinary skill in the art” is relevant, Appellants have shown by way of specific reference to the systems of Anderson and Duruoz as a whole that one of ordinary skill in the art would not be led to combine Anderson and Duruoz.

Since, as described above, Appellants respectfully disagree that Anderson can be combined with Duruoz in the manner suggested by the invention due to lack of interchangeability of the pre-defined DSP Task Unit structure of Anderson and the command buffers of Duruoz, Appellants respectfully submit that Anderson and Duruoz cannot be combined in the manner suggested to teach or suggest a method for “controlling the at least one coprocessor of the computing system with commands of command buffers submitted to the at least one coprocessor by a host processor of the host computing system, transmitting, by the at least one coprocessor, data back to the host computing system in response to commands in at least one command buffer of the command buffers; and scheduling the transmission of the commands of the command buffers by a managing object included in the host computing system...,” as recited in claim 1.

In addition, Appellants respectfully submit that Anderson and Duruoz cannot be combined in the manner suggested to teach or suggest a computer readable medium having “a managing object for controlling the at least one coprocessor of the computing system with command data of command buffers submitted to the at least one coprocessor by a host processor of the host computing system and for scheduling the transmission of the command data and means for transmitting, by the at least one coprocessor, data back to the host computing system in response to command data of at least one command buffer of the command buffers,” as recited in claim 27.

Appellants also respectfully submit that Anderson and Duruoz cannot be combined in the manner suggested to teach or suggest a computing device having “a managing object for

controlling the at least one coprocessor of the host computing system with command data of command buffers submitted to the at least one coprocessor by the host processor of the host computing system and for scheduling the transmission of the command data of the command buffers and means for transmitting, by the at least one coprocessor, data back to the host computing system in response to command data in at least one command buffer of the command buffers,” as recited in claim 27.

Accordingly, Appellants respectfully request reconsideration of the rejection under 35 U.S.C. § 103 of the method of claims 1-13 and 16-23, the computer readable media of claims 27-39 and 42-49 and the computing device recited in claims 52-64 and 67-74.

D. Appellants Detailed Specification Concerning Command Buffer Scheduling Provides Further Evidence that the Invention is not Obvious.

Further underscoring the non-obviousness of Appellants’ invention is Appellants’ own detailed description describing in various non-limiting embodiments the many intricacies of implementing command buffer scheduling in modern graphics architectures, which are not disclosed, taught or suggested by Anderson or Duruoz. For instance, as described in the exemplary, non-limiting implementation found at Fig. 5 at page 37, line 8 to page 40, line 12,

Fig. 5 clarifies the sequence of events that occur when an application is making API calls to perform graphics operations. The block diagram components of Fig. 5 are shading coded: user mode components are of medium shade, kernel mode components are shaded lightly, and the hardware is shaded darkly. The command buffers are not specifically depicted in Fig. 5 as a hardware component; as per Fig. 4, the user mode driver 520 writes hardware-specific commands into the device’s current command buffer, the command buffer scheduler (part of the system kernel support 530) dispatches the command buffer to the hardware 560 via the kernel mode driver 550, and finished command buffers are recycled for use by an application in the system. It is noted that multiple applications 500 can potentially share the pool of available command buffers, with the system kernel support 530 arbitrating sharing of that resource.

When the application 500 initially creates a drawing context 501, the system kernel support 530 checks to see whether a new command buffer can be created 531. If so, the new command buffer is created 532 and initialized 533, and the thread obtains an initialized command buffer 534 before the Application 500 can perform drawing calls 502. If a command buffer could not be created in step 531, the application 500 must wait until an initialized command buffer becomes available 534. Once the application 500 has obtained a command buffer, the application 500, runtime 510 and user mode driver 520 enter the typical interaction between the three components that

cause hardware-specific commands to be written into the command buffer. The drawing calls 502 from the application 500 are validated 511 by the runtime 510; a check 512 then determines whether a flush of the current command buffer is needed. If not, the drawing command is translated to a simpler, canonical DDI call 513 and passed to the user mode driver 520. The driver translates the DDI call into hardware specific commands and attempts to write them into the command buffer. If the check 522 for flush determines that there is no room in the command buffer, the command buffer must be submitted to the system kernel support 530 and a new command buffer obtained from same before the command can be written and execution can continue. If either the runtime 510 or the user mode driver 520 determines that a flush is needed, per step 535 the command buffer is added to the waiting queue. At that time, the system kernel can check 536 whether the command buffer can be submitted forthwith (typically because no command buffer is running). If not, the command buffer is left in the waiting queue and a new command buffer must be obtained 534. It is noted that this functional block, which waits until a suitable initialized command buffer is available and then allocates it to the device, is identical to the operation needed by the application 500 before it can begin drawing.

The arrows are also shading coded. The darkest arrows trace the “typical” thread execution of the application 500 as it initializes and performs drawing commands and the runtime 510 and user mode driver 520 translate those drawing commands into hardware-specific commands and write them into the command buffer. If the runtime 510 or user mode driver 520 determine that a flush is needed (i.e., the command buffer must be submitted), the thread of execution designated by green arrows is followed: the command buffer is added to the waiting queue and submitted if possible, then a new command buffer is obtained so execution can continue.

The medium shaded arrows trace the “typical” path taken by a command buffer after it is added to the waiting queue in step 535. If the command buffer can be dispatched immediately (the check 536), the command buffer is marked as ready 537 and selected for dispatch 540. Otherwise, the waiting command buffer must be marked Ready in the normal course of events, when the current running command buffer finishes execution.

When a ready command buffer is selected for dispatch 540, the system kernel support 530 has the kernel driver 550 context switch the hardware to the appropriate context 551 and dispatch the command buffer to the hardware 552. The hardware then reads and executes the command buffer 561, until it is preempted or the command buffer finishes. If the command buffer completes normally 563, the hardware signals an interrupt and the interrupt service routine 553 executes. The ISR may wish to save the hardware context 554 at this time, although the driver may wish to defer this operation to the context switch 551, in case the hardware should be asked to execute two command buffers in a row that operate on the same context. After this step 554, the kernel system support 530 can free the resources needed by that command buffer 538, as well as signal any notification mechanisms such as events to let interested clients know that the command buffer is completed. After step 538,

the kernel system support has two distinct tasks: it must reinitialize the newly available command buffer and add it to the initialized pool 533, and it must unblock any waiting command buffers and move them into the ready queue 539. After step 539, another command buffer can be selected for dispatch 540. The case of preemption is handled slightly differently than normal command buffer completion, as delineated by the orange arrows. Since the preempted command buffer did not finish, the system must not notify clients of its completion (step 538) or unblock dependent command buffers (step 539). Instead, the driver saves the context 554 of the partially executed command buffer such that it can be restarted where the preemption occurred, notifies any clients 541 that need to know about the preemption, and selects the next Ready command buffer to dispatch 540.

Examples of occurrences that might cause preemption include external events such as VSYNC or the arrival of the display refresh to a particular location such as a scanline, or expiration of a time quantum. In addition, for management of time quanta, the operating system may use a mechanism to preempt the graphics processor by calling the Kernel Driver 550. This would equate to expiration of a time quantum as arbitrated by the hardware, except that the complexity and responsibility for deciding when to preempt the graphics processor then rests with the System Kernel Support 530 rather than the Hardware 560. It is presumed that the system has previously set up a response to such an event, such as executing a series of Blt commands when the refresh passes a certain scanline (for a tear-free update).

Note that many of the ideas presented above can be applied to the system described in connection with Figs 4 and 5. For example, per the above, the system kernel support 540 could potentially take over the context switching task 551 by having the kernel driver 550 dispatch a context switch command buffer. For another example, lightweight contexts could be supported in this system by differentiating between context types and, potentially, types of command buffers, as described in further detail above. For another example, step 540 ("Select ready command buffer for dispatch") could be modified slightly to take advantage of the ideas presented above, in which the hardware begins executing a previously-selected command buffer as soon as the running command buffer finishes. For yet another example, an efficient way for the runtime 510 and user mode Driver 520 to detect command buffer overflow 522 would be for the system kernel support 530 to allocate an unwritable guard page at the end of each command buffer, and use structured exception handling to detect when a write to the guard page has been attempted. Finally, the ideas of making the system robust in the face of invalid command buffers could be readily incorporated into this system, either by signaling an exception when an invalid command caused the hardware to generate an interrupt, or by resetting the hardware via a special kernel driver 550 entry point if it became unresponsive for too long.

Fig. 5 is only one embodiment of the invention. The detailed description also describes how to, for instance, implement the command buffer memory of the command buffer scheduling of the invention so that it is made visible to user mode, kernel mode, or

both, describes numerous options as to the nature of the executable code that is writing into the receiving command buffer, and describes how to include a JIT (Just-In-Time) compilation component that can translate a hardware-independent intermediate language into object code for execution by the host microprocessor. Figs. 3A through 3E depict several of these options, i.e., several different architectures for writing into the command buffer. Page 23, line 21 to page 25, line 24.

Appellants thus respectfully submit that such complexities and intricacies of implementing scheduling of command buffers in modern graphics architectures, as described in Appellants' specification, are strong evidence suggesting that one of ordinary skill in the art would not know how to modify Anderson to include Duruoz's command buffers, based on the disclosures of Anderson and Duruoz alone, and that such modifications would be non-trivial and non-obvious to one of ordinary skill in the art at the time of Appellants' invention.

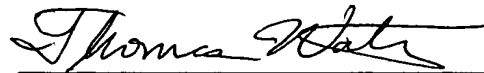
2. Claims 14-15, 40-41 and 65-66 are not obvious over Anderson in view of Duruoz and further in view of Hendler.

To the extent that the rejection to claims 14-15, 40-41 and 65-66 is based on the combination of Anderson and Duruoz, Appellants respectfully submit that the combination of Anderson, Duruoz and Hendler is also improper for the same reasons as with respect to claims 1-13, 16-23, 27-39, 42-49, 52-64 and 67-74. If Anderson and Duruoz are improperly combinable due to lack of substitutability between Anderson's Task Units and Duruoz's command buffers, then the combination of Anderson, Duruoz and Hendler is also believed to be improper for at least the same reasons. Reconsideration and withdrawal of the rejection to claims 14-15, 40-41 and 65-66 is thus respectfully requested.

3. Conclusion of Argument

For the foregoing reasons, Appellants respectfully submit that the claims on appeal are patentable over the art of record. Appellants thus request that the Board reverse the rejection of the claims on appeal.

Respectfully submitted,



Date: May 16, 2006

Thomas E. Watson
Registration No. 43,243

Woodcock Washburn LLP
One Liberty Place - 46th Floor
Philadelphia PA 19103
Telephone: (215) 568-3100
Facsimile: (215) 568-3439